

# 라즈베리파이 먼지센서 측정 Raspberry Pi dust sensor measurement

기말 프로젝트

충북대학교 산업인공지능학과 이민수

1. 프로젝트 개요
2. 프로젝트 결과

# 1. 프로젝트 개요

## ■ 프로젝트 개요

- **프로젝트명** : 라즈베리파이 먼지센서 측정

Raspberry Pi dust sensor measurement

- **내용요약**

- 먼지센서(PMS7000)와 부유입자측정기(Particle Counter)와의 data 확인
- Data 비교를 통하여 먼지센서(PMS7000) 공정 모니터링 사용 판단

- **방법**

- 라즈베리파이 및 먼지센서(PMS7000) 연결
- 먼지센서(PMS7000)와 부유입자측정기(Particle Counter) 환경 조건을 동일시하여 측정

# 1. 프로젝트 개요

## ■ 프로젝트 개요

### • 주제선정

- Cleanroom ISO 5 Class는 매 작업시마다 Particle Counter(부유입자)를 측정 해야 한다 (ISO 14644에 근거함)
- Particle Counter는 현재 1대 운용중이 있으며, 모니터링 공정과 ISO 5 공정이 동시에 진행 될 수 없기에 2024년 1대 추가로 구비할 예정이다
- Particle Counter의 가격은 약 1,700 만원 이다
- 만약 PMS7000과 Particle Counter간의 data의 차이가  $\pm 10\%$  이내라고 판단될 경우 많은 비용을 절약할 수 있을 것이라 판단 된다

# 1. 프로젝트 개요

## ■ 프로젝트 개요

### • 추진일정

주요 추진 내용	12	13	14	15
주제선정 및 장치구성				
장치구매 및 실험환경 조성				
실험결과 정리				
발표자료 작성				

### • 세부일정

- 주제선정 및 장치구성 : 현업에서 필요하며 적용가능한 주제 선정/라즈베리파이 및 미세먼지 측정에 필요한 소모품(PMS7000)
- 장치구매 및 실험환경 조성 : 온라인구매/실험환경은 비클린룸에서 진행
- 실험결과 정리 : PMS7000으로 측정한 미세먼지 입자 크기와 Particle Counter간 data를 비교 정리
- 발표자료 작성 : 위 내용은 근거로 자료 작성

# 1. 프로젝트 개요

## ■ 프로젝트 개요

### • 장치설명

NO.	장치명	설명
1	라즈베리파이	영구의 라즈베리 파이 재단에서 기초 컴퓨터 과학 교육을 목적으로 개발한 Single board computer
2	PMS 7003 먼지 센서	디지털 범용 입자 농도 센서
3	먼지센서 인터페이스 보드	
4	USB to Usart 변환케이블	

# 1. 프로젝트 개요

## ■ 프로젝트 개요

### • 조립모습



# 1. 프로젝트 개요

## ■ 프로젝트 결과

### • 프로그램연결

```
pi@raspberrypi: ~/PMS7003
pi@raspberrypi:~$ git clone https://github.com/eleparts/PMS7003
Cloning into 'PMS7003'...
remote: Counting objects: 24, done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 24 (delta 12), reused 21 (delta 10), pack-reused 0
Unpacking objects: 100% (24/24), done.
pi@raspberrypi:~$ ll
total 48
drwxr-xr-x 2 pi pi 4096 Jun 27 02:22 Desktop
drwxr-xr-x 2 pi pi 4096 Jun 27 02:22 Documents
drwxr-xr-x 2 pi pi 4096 Jun 27 02:22 Downloads
drwxr-xr-x 2 pi pi 4096 Jun 27 02:00 MagPi
drwxr-xr-x 2 pi pi 4096 Jun 27 02:22 Music
drwxr-xr-x 3 pi pi 4096 Aug 24 09:05 oldconffiles
drwxr-xr-x 2 pi pi 4096 Jun 27 02:22 Pictures
drwxr-xr-x 4 pi pi 4096 Aug 27 03:26 PMS7003
drwxr-xr-x 2 pi pi 4096 Jun 27 02:22 Public
drwxr-xr-x 2 pi pi 4096 Jun 27 01:59 python_games
drwxr-xr-x 2 pi pi 4096 Jun 27 02:22 Templates
drwxr-xr-x 2 pi pi 4096 Jun 27 02:22 Videos
pi@raspberrypi:~$ cd PMS7003/
pi@raspberrypi:~/PMS7003$
```

```
GNU nano 2.7.4 File:
print ("Reserved F : %s | Reserved B
print ("CHKSUM : %s | read CHKSUM :
print ("=====

# UART / USB Serial : 'dmesg | grep ttyU
USB0 = '/dev/ttyUSB0'
UART = '/dev/ttyAMA0'

# USE PORT
SERIAL_PORT = UART

# Baud Rate
Speed = 9600

# example
if __name__=='__main__':
```

```
pi@raspberrypi: ~/PMS7003
GNU nano 2.7.4 File:
print ("Reserved F : %s | Reserved B
print ("CHKSUM : %s | read CHKSUM :
print ("=====

# UART / USB Serial : 'dmesg | grep ttyU
USB0 = '/dev/ttyUSB0'
UART = '/dev/ttyAMA0'

# USE PORT
SERIAL_PORT = USB0

# Baud Rate
Speed = 9600

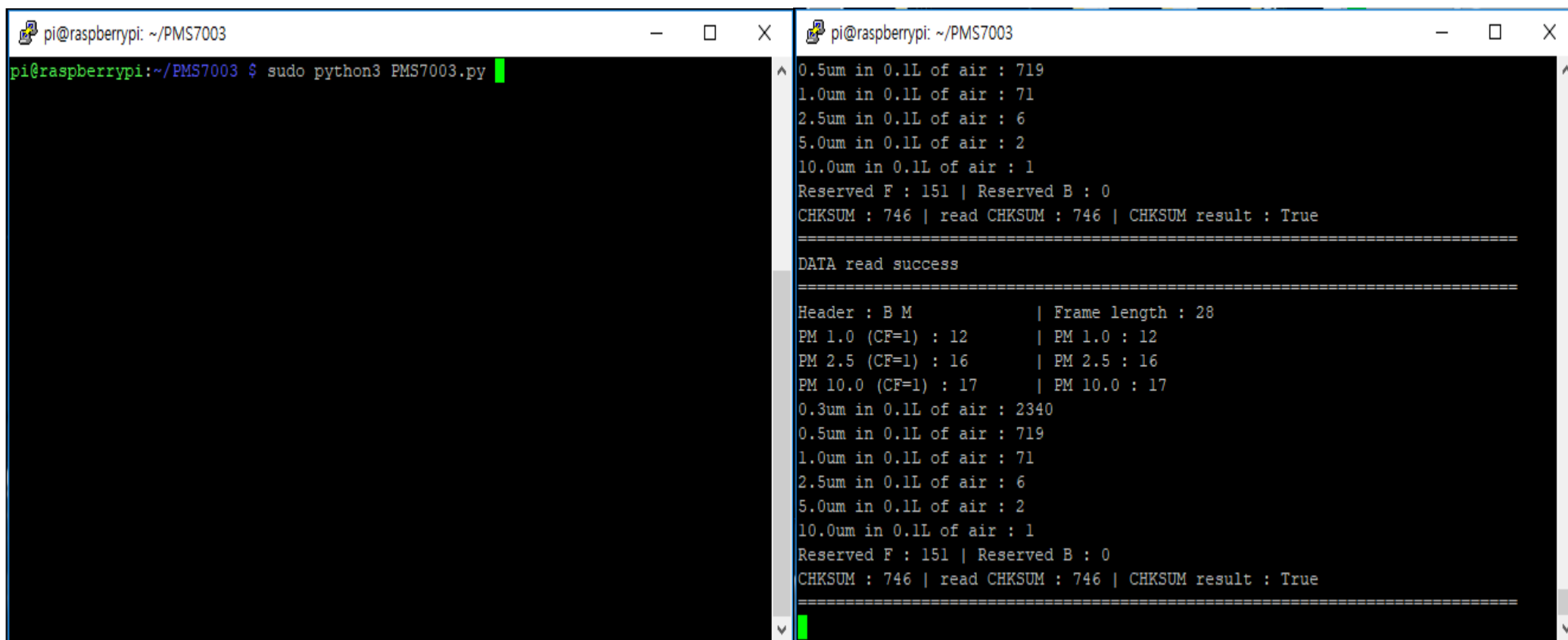
# example
if __name__=='__main__':
```



# 1. 프로젝트 개요

## 프로젝트 결과

- 프로그램연결



```
pi@raspberrypi: ~/PMS7003
pi@raspberrypi:~/PMS7003 $ sudo python3 PMS7003.py

0.5um in 0.1L of air : 719
1.0um in 0.1L of air : 71
2.5um in 0.1L of air : 6
5.0um in 0.1L of air : 2
10.0um in 0.1L of air : 1
Reserved F : 151 | Reserved B : 0
CHKSUM : 746 | read CHKSUM : 746 | CHKSUM result : True
=====
DATA read success
=====
Header : B M          | Frame length : 28
PM 1.0 (CF=1) : 12     | PM 1.0 : 12
PM 2.5 (CF=1) : 16     | PM 2.5 : 16
PM 10.0 (CF=1) : 17    | PM 10.0 : 17
0.3um in 0.1L of air : 2340
0.5um in 0.1L of air : 719
1.0um in 0.1L of air : 71
2.5um in 0.1L of air : 6
5.0um in 0.1L of air : 2
10.0um in 0.1L of air : 1
Reserved F : 151 | Reserved B : 0
CHKSUM : 746 | read CHKSUM : 746 | CHKSUM result : True
=====
```

# 1. 프로젝트 개요

## ■ 프로젝트 결과

### • 코드

```
import serial
import struct
import time
```

```
class PMS7003(object):
```

```
    # PMS7003 protocol data (HEADER 2byte + 30byte)
    PMS_7003_PROTOCOL_SIZE = 32
```

```
    # PMS7003 data list
```

```
    HEADER_HIGH      = 0 # 0x42
    HEADER_LOW       = 1 # 0x4d
    FRAME_LENGTH     = 2 # 2x13+2(data+check bytes)
    DUST_PM1_0_CF1    = 3 # PM1.0 concentration unit  $\mu$ g/m3 (CF=1, standard particle)
    DUST_PM2_5_CF1    = 4 # PM2.5 concentration unit  $\mu$ g/m3 (CF=1, standard particle)
    DUST_PM10_0_CF1   = 5 # PM10 concentration unit  $\mu$ g/m3 (CF=1, standard particle)
    DUST_PM1_0_ATM    = 6 # PM1.0 concentration unit  $\mu$ g/m3 (under atmospheric environment)
    DUST_PM2_5_ATM    = 7 # PM2.5 concentration unit  $\mu$ g/m3 (under atmospheric environment)
    DUST_PM10_0_ATM   = 8 # PM10 concentration unit  $\mu$ g/m3 (under atmospheric environment)
    DUST_AIR_0_3      = 9 # indicates the number of particles with diameter beyond 0.3  $\mu$ m in 0.1 L of air.
    DUST_AIR_0_5      = 10 # indicates the number of particles with diameter beyond 0.5  $\mu$ m in 0.1 L of air.
    DUST_AIR_1_0      = 11 # indicates the number of particles with diameter beyond 1.0  $\mu$ m in 0.1 L of air.
    DUST_AIR_2_5      = 12 # indicates the number of particles with diameter beyond 2.5  $\mu$ m in 0.1 L of air.
    DUST_AIR_5_0      = 13 # indicates the number of particles with diameter beyond 5.0  $\mu$ m in 0.1 L of air.
    DUST_AIR_10_0     = 14 # indicates the number of particles with diameter beyond 10  $\mu$ m in 0.1 L of air.
    RESERVEDF        = 15 # Data13 Reserved high 8 bits
    RESERVEDB        = 16 # Data13 Reserved low 8 bits
    CHECKSUM         = 17 # Checksum code
```

```
    # header check
    def header_chk(self, buffer):
```

```
        if (buffer[self.HEADER_HIGH] == 66 and buffer[self.HEADER_LOW] == 77):
            return True
```

```
        else:
            return False
```

```
    # chksum value calculation
    def chksum_cal(self, buffer):
```

```
        buffer = buffer[0:self.PMS_7003_PROTOCOL_SIZE]
```

```
        # data unpack (Byte -> Tuple (30 x unsigned char <B> + unsigned short <H>))
        chksum_data = struct.unpack('!30BH', buffer)
```

```
        chksum = 0
```

```
        for i in range(30):
            chksum = chksum + chksum_data[i]
```

```
        return chksum
```

```
    # checksum check
    def chksum_chk(self, buffer):
```

```
        chk_result = self.chksum_cal(buffer)
```

```
        chksum_buffer = buffer[30:self.PMS_7003_PROTOCOL_SIZE]
        chksum = struct.unpack('!H', chksum_buffer)
```

```
        if (chk_result == chksum[0]):
            return True
```

```
        else:
            return False
```

# 1. 프로젝트 개요

## 프로젝트 결과

### 코드

```
# protocol size(small) check
def protocol_size_chk(self, buffer):

    if(self.PMS_7003_PROTOCOL_SIZE <= len(buffer)):
        return True

    else:
        return False

# protocol check
def protocol_chk(self, buffer):

    if(self.protocol_size_chk(buffer)):

        if(self.header_chk(buffer)):

            if(self.chksum_chk(buffer)):

                return True
            else:
                print("Chksum err")
        else:
            print("Header err")
    else:
        print("Protol err")

    return False

# unpack data
# <Tuple (13 x unsigned short <H> + 2 x unsigned char <B> + unsigned short <H>)>
def unpack_data(self, buffer):
```

```
# unpack data
# <Tuple (13 x unsigned short <H> + 2 x unsigned char <B> + unsigned short <H>)>
def unpack_data(self, buffer):

    buffer = buffer[0:self.PMS_7003_PROTOCOL_SIZE]

    # data unpack (Byte -> Tuple (13 x unsigned short <H> + 2 x unsigned char <B> + unsigned short <H>))
    data = struct.unpack('!2B13H2BH', buffer)

    return data

def print_serial(self, buffer):

    chksum = self.chksum_cal(buffer)
    data = self.unpack_data(buffer)

    print ("=====")
    print ("Header : %c %c \t\t | Frame length : %s" % (data[self.HEADER_HIGH], data[self.HEADER_LOW], data[self.FRAME_LENGTH]))
    print ("PM 1.0 (CF=1) : %s\t | PM 1.0 : %s" % (data[self.DUST_PM1_0_CF1], data[self.DUST_PM1_0_ATM]))
    print ("PM 2.5 (CF=1) : %s\t | PM 2.5 : %s" % (data[self.DUST_PM2_5_CF1], data[self.DUST_PM2_5_ATM]))
    print ("PM 10.0 (CF=1) : %s\t | PM 10.0 : %s" % (data[self.DUST_PM10_0_CF1], data[self.DUST_PM10_0_ATM]))
    print ("0.3um in 0.1L of air : %s" % (data[self.DUST_AIR_0_3]))
    print ("0.5um in 0.1L of air : %s" % (data[self.DUST_AIR_0_5]))
    print ("1.0um in 0.1L of air : %s" % (data[self.DUST_AIR_1_0]))
    print ("2.5um in 0.1L of air : %s" % (data[self.DUST_AIR_2_5]))
    print ("5.0um in 0.1L of air : %s" % (data[self.DUST_AIR_5_0]))
    print ("10.0um in 0.1L of air : %s" % (data[self.DUST_AIR_10_0]))
    print ("Reserved F : %s | Reserved B : %s" % (data[self.RESERVEDF], data[self.RESERVEDB]))
    print ("CHKSUM : %s | read CHKSUM : %s | CHKSUM result : %s" % (chksum, data[self.CHECKSUM], chksum == data[self.CHECKSUM]))
    print ("=====")
```

# 1. 프로젝트 개요

## ■ 프로젝트 결과

### • 코드

```
# UART / USB Serial : 'dmesg | grep ttyUSB'
USB0 = '/dev/ttyUSB0'
UART = '/dev/ttyAMA0'

# USE PORT
SERIAL_PORT = UART

# Baud Rate
Speed = 9600

# example
if __name__=='__main__':

    #serial setting
    ser = serial.Serial(SERIAL_PORT, Speed, timeout = 1)

    dust = PMS7003()

    while True:

        ser.flushInput()
        buffer = ser.read(1024)

        if(dust.protocol_chk(buffer)):

            print("DATA read success")

            # print data
            dust.print_serial(buffer)

        else:

            print("DATA read fail...")

    ser.close()
```

# 1. 프로젝트 개요

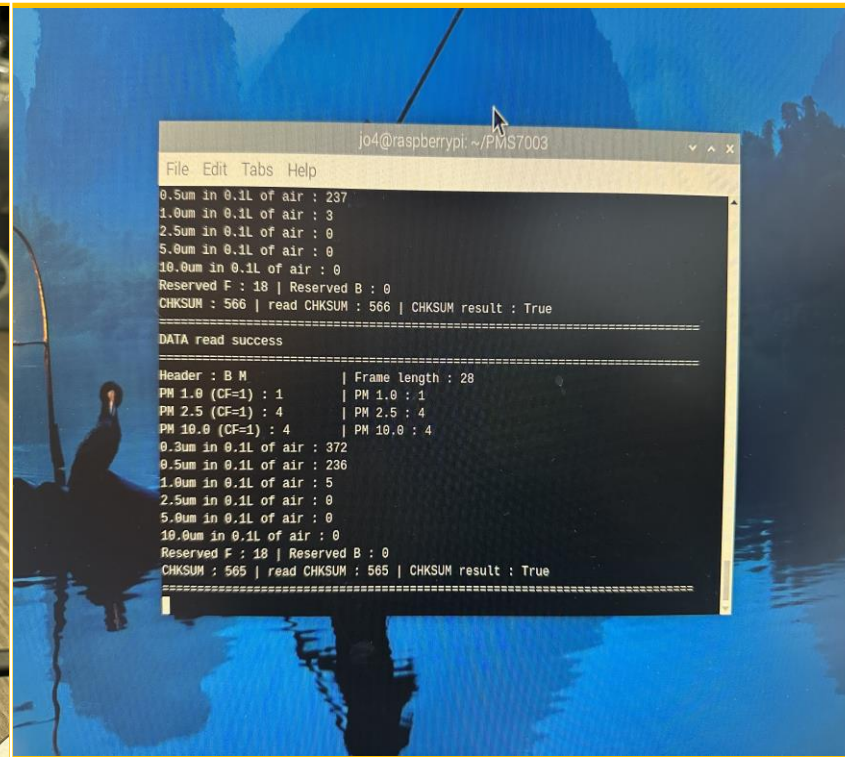
## ■ 프로젝트 결과

### • Data 측정 예시

**Particle Counter**



**PMS 7003**



# 1. 프로젝트 개요

## ■ 프로젝트 결과

### • Data 비교

Particle Counter					PMS 7003			
G	H	I	J	K	I	J	K	L
Ch1 0.3μm	Ch2 0.5μm	Ch3 1μm	Ch4 3μm	Ch5 5μm	#	Ch1 0.3μm	Ch2 0.5μm	Ch3 1μm
39,206,720	10,989,610	3,829,930	110,318	9,399	1	196,119	156,782	21,225
39,468,090	10,644,590	3,535,124	89,894	6,749	2	196,968	157,631	21,508
40,324,660	10,731,660	3,530,071	90,106	5,730	3	212,816	174,328	24,621
39,896,260	10,418,900	3,306,396	76,714	6,148	4	213,665	175,743	24,338
39,968,300	10,066,930	3,039,258	68,587	4,947	5	209,986	173,196	27,168

## ■ 프로젝트 결과

### • 결론

- Particle Counter와 PMS 7003의 Data간 차이는  $0.3\mu\text{m}$  약 200배,  $0.5\mu\text{m}$  약 70배 차이가 발생하는 것을 알 수 있다
- 차이가 발생하는 원인으로는 공기 흡입량 차이로 판단 된다 Particle Counter는 1분간 28.3L를 흡입하는 반면 PMS 7003은 1초에 0.1L의 공기를 흡입한다
- 공집 흡입량부터 283배 차이는 매우 큰 숫자이며 이를 값에 비례하여 산출한다고 하더라도 결과값에 큰 영향을 미치는것 같다
- 또한 PMS 7003의 교정이 불가능하다는 교정 업체 의견에 따라 아래와 같이 결론을 도출하였다
- PMS 7003으로 Particle Counter의 부유입자 실험을 대체할 수 없다.  
(Validation이 아닌 모니터링 조차도 값의 차이가 너무 크게 발생)

# Thank You!